Project October 2025

Natasha Fontanilla

2025-11-05

Contents

Building a DOCKER **3**

Download from GitHub **4**

OVERWIEW **4**

TASK 1 **4**

DATA.TABLE VERSION 4

DATA.FRAME VERSION 6

TASK 2 **8**

DATA.TABLE VERSION 8

TASK 3 **9**

DATA.TABLE VERSION 9

DATA.FRAME VERSION 10

TASK 4 **11**

DATA.TABLE VERSION 11

DATA.FRAME VERSION 13

TASK 5 **14**

DATA.TABLE VERSION 14

DATA.FRAME VERSION 15

TASK 6 **17**

DATA.TABLE VERSION 17

DATA.FRAME VERSION 18

TASK 7 **18**
DATA.TABLE VERSION 18
DATA.FRAME VERSION 19

TASK 8 **21**
DATA.TABLE VERSION 21
DATA.FRAME VERSION 22

TASK 9 **24**
DATA.TABLE VERSION 24
DATA.FRAME VERSION 27

TASK 10 **29**
DATA.TABLE VERSION 30
DATA.FRAME VERSION 32

TASK 11 **34**
DATA.TABLE VERSION 34
DATA.FRAME VERSION 35

TASK 12 **37**
DATA.TABLE VERSION 37
DATA.FRAME VERSION 39

FINAL REVISION **41**
DATA.TABLE VERSION 41
DATA.FRAME VERSION 45

Building a DOCKER

The first step is to create a project folder, which will contain all the files needed to build the container. Inside this folder, a file named Dockerfile will be created. The Dockerfile defines the environment and the instructions for building the container.

To create the Dockerfile from the terminal (bash), use:

Once opened, the Dockerfile can be filled with the required configuration. For this specific project, it was required to install Rstudio inside the docker where all the analysis will be performed.

```
# Start from Ubuntu
FROM ubuntu:22.04

# Avoid prompts during install
ENV DEBIAN_FRONTEND=noninteractive

# Install R and dependencies needed
RUN apt-get update && apt-get install -y \
    r-base \
    gdebi-core \
    wget \
    sudo \
    && apt-get clean

# Install RStudio Server
RUN wget https://download2.rstudio.org/server/jammy/amd64/rstudio-server-2024.04.2-764-amd64.deb -O /tm
    gdebi -n /tmp/rstudio-server.deb && \
    rm /tmp/rstudio-server.deb

# Create RStudio user and password
RUN useradd -m rstudio && echo "rstudio:rstudio" | chpasswd && adduser rstudio sudo

# Install essential R packages
RUN R -q -e "install.packages(c('devtools'), repos='https://cloud.r-project.org')"
RUN R -q -e "install.packages(c('data.table', 'dplyr', 'ggplot2'), repos='https://cloud.r-project.org')

# Set working directory
WORKDIR /home/project_oct

# Expose RStudio Server port
EXPOSE 8787

# Default command: start RStudio Server
CMD ["/usr/lib/rstudio-server/bin/rserver", "--server-daemonize=0"]
```

To complete the building of the container, open the terminal in the folder containing the Dockerfile and run:

```
{bash echo=FALSE, eval=FALSE} docker build -t my_rstudio .
```

To run the Docker:

docker run -d -p 8787:8787 -v $(pwd):/home/project_oct –name project_container my_rstudio

After running, R studio will open and will require a Username and a password defined in the Dockerfile (rstudio). The Rstudio will open.

Download from GitHub

For this specific project, a custom package called "NF" was created and needs to be downloaded.

In the CONSOLE of R write:

```
install.packages("remotes")
remotes::install_github("natashafon/bioinfo_project_oct25")
```

Load the package:

```
library(NF)
```

```
ls("package:NF")
```

Once the Docker container is running and the repository is installed, the RStudio environment will display a folder named bioinfo_project_oct25. This directory contains all the components required for the project:

- R — contains all the functions included in the custom NF package.
- Rscripts — contains the original analysis scripts from which the final functions were derived.
- R markdown
- doc -.gitignore
- Dockerfile
- LICENSE
- README.md

Change the directory to begin the analysis.

```
setwd("~/bioinfo_project_oct25/data")
```

OVERWIEW

The project is built around data.table package and data.frame R structure. There are 12 tasks and each one will have a data.table and data.frame version.

TASK 1

The objective of this task is to filter, summarize, and group bulk RNA-seq count data. The analysis uses two input files: bulk_counts_long.csv, which contains gene expression counts per sample, and sample_metadata.csv, which includes sample annotations. The dataset is first filtered to retain only the samples classified as "treated" and the genes whose names begin with "GENE_00". For each gene, the mean and median expression values are then calculated. Next, the filtered count data is joined with the sample metadata to compute per-condition mean expression values in a single, streamlined workflow.

DATA.TABLE VERSION

```r
task1_dt <- function(counts_file = "bulk_counts_long.csv",
                     meta_file   = "sample_metadata.csv") {
# Load libraries
  library(data.table)

# Read th CSV files
counts <- fread(counts_file)   # columns: gene, sample_id, count
meta   <- fread(meta_file)    # columns: sample_id, condition, ...

# Ensure sample_id is character
counts[, sample_id := as.character(sample_id)]
meta[,   sample_id := as.character(sample_id)]

# Join on sample_id
merged <- merge(counts, meta, by = "sample_id", sort = FALSE)

# Keep only treated samples and genes starting with "GENE_00"
filtered <- merged[
  condition == "treated" & grepl("^GENE_00", gene)
]


# Mean and median count by gene
gene_summary <- filtered[, .(
  mean_count   = mean(count, na.rm = TRUE),
  median_count = median(count, na.rm = TRUE)
), by = gene][order(gene)]


# Per-condition mean counts by gene
per_condition_mean <- merged[, .(
  mean_count = mean(count, na.rm = TRUE)
), by = .(gene, condition)][order(gene, condition)]


return(list(
  merged = merged,
  filtered = filtered,
  gene_summary = gene_summary,
  per_condition_mean = per_condition_mean
))
}

# SHOW RESULTS
res_dt <- task1_dt()
head(res_dt$filtered, 5)
```

```
##    sample_id      gene count condition  batch patient_id timepoint
##       <char>    <char> <int>    <char> <char>     <char>    <char>
## 1:       S01 GENE_0049     9   treated batch1       P014        T0
## 2:       S01 GENE_0054    10   treated batch1       P014        T0
## 3:       S01 GENE_0018    17   treated batch1       P014        T0
## 4:       S01 GENE_0004     4   treated batch1       P014        T0
```

```
## 5:       S01 GENE_0070     8   treated batch1       P014        T0
```

```
head(res_dt$gene_summary, 5)
```

```
##         gene mean_count median_count
##       <char>      <num>        <num>
## 1: GENE_0000   7.066667            6
## 2: GENE_0001   2.533333            2
## 3: GENE_0002   3.800000            3
## 4: GENE_0003  20.266667           19
## 5: GENE_0004   5.933333            7
```

```
head(res_dt$per_condition_mean, 5)
```

```
##         gene condition mean_count
##       <char>    <char>      <num>
## 1: GENE_0000   control   6.111111
## 2: GENE_0000   treated   7.066667
## 3: GENE_0001   control   2.666667
## 4: GENE_0001   treated   2.533333
## 5: GENE_0002   control   4.000000
```

DATA.FRAME VERSION

```
task1_df <- function(counts_file = "bulk_counts_long.csv",
                     meta_file   = "sample_metadata.csv") {

# Read CSV files
counts_df <- read.csv(counts_file, stringsAsFactors = FALSE)
meta_df   <- read.csv(meta_file,  stringsAsFactors = FALSE)

# Ensure sample_id is character
counts_df$sample_id <- as.character(counts_df$sample_id)
meta_df$sample_id   <- as.character(meta_df$sample_id)

# Join on sample_id
merged_df <- merge(counts_df, meta_df, by = "sample_id", sort = FALSE)

# Filter treated + genes starting with "GENE_00"
filtered_df <- merged_df[
  merged_df$condition == "treated" & grepl("^GENE_00", merged_df$gene),
]

# Mean and median by gene (on filtered)
mean_by_gene   <- aggregate(count ~ gene, data = filtered_df,
                            FUN = function(x) mean(x, na.rm = TRUE))
median_by_gene <- aggregate(count ~ gene, data = filtered_df,
                            FUN = function(x) median(x, na.rm = TRUE))

# Merge summaries and align column names/order to data.table version
```

```r
gene_summary_df <- merge(mean_by_gene, median_by_gene, by = "gene")

names(gene_summary_df) <- c("gene", "mean_count", "median_count")

gene_summary_df <- gene_summary_df[order(gene_summary_df$gene), ]

# Per-condition mean counts by gene (on all merged)
per_condition_mean_df <- aggregate(
  count ~ gene + condition,
  data = merged_df,
  FUN = function(x) mean(x, na.rm = TRUE)
)
per_condition_mean_df <- per_condition_mean_df[order(
  per_condition_mean_df$gene, per_condition_mean_df$condition), ]

return(list(
  merged_df = merged_df,
  filtered_df = filtered_df,
  gene_summary_df = gene_summary_df,
  per_condition_mean_df = per_condition_mean_df
))
}

# SHOW RESULTS
res_df <- task1_df()
head(res_df$filtered_df, 5)
```

```
##    sample_id      gene count condition  batch patient_id timepoint
## 5        S01 GENE_0049     9   treated batch1       P014        T0
## 9        S01 GENE_0054    10   treated batch1       P014        T0
## 12       S01 GENE_0018    17   treated batch1       P014        T0
## 16       S01 GENE_0004     4   treated batch1       P014        T0
## 24       S01 GENE_0070     8   treated batch1       P014        T0
```

```r
head(res_df$gene_summary_df, 5)
```

```
##        gene mean_count median_count
## 1 GENE_0000   7.066667            6
## 2 GENE_0001   2.533333            2
## 3 GENE_0002   3.800000            3
## 4 GENE_0003  20.266667           19
## 5 GENE_0004   5.933333            7
```

```r
head(res_df$per_condition_mean_df, 5)
```

```
##          gene condition    count
## 1   GENE_0000   control 6.111111
## 501 GENE_0000   treated 7.066667
## 2   GENE_0001   control 2.666667
## 502 GENE_0001   treated 2.533333
## 3   GENE_0002   control 4.000000
```

TASK 2

The goal of this task is to create quality-control style derived variables directly within the count dataset, without producing unnecessary copies of the data in memory. The analysis is based on the file bulk_counts_long.csv, which contains gene expression counts for each sample. First, a log -transformed count column is added to the dataset. A binary indicator column (high) is then created, which initially flags samples with counts greater than 100. In the second step, this indicator is refined to be gene-specific: for each gene, high is reassigned based on whether a sample's count is greater than the gene-wise median count.

DATA.TABLE VERSION

```
task2_dt <- function(
    counts_file = "bulk_counts_long.csv") {

# Load the library
library(data.table)

# Read the CSV file
counts_dt <- fread(counts_file)

# Adding log2 counts coulumn which is applied to all rows
counts_dt[, log2_count := log2(count + 1)]

# Adding a binary flag high if count >100
counts_dt[, high := count > 100]

# Overwrite 'high' if the count is higher thant the median value for a certain gene
counts_dt[, high := count > median(count), by = gene]

return(counts_dt)
}

# SHOW RESULTS
res_dt2 <- task2_dt("bulk_counts_long.csv")
head(res_dt2, 5)
```

```
##         gene sample_id count log2_count   high
##       <char>    <char> <int>      <num> <lgcl>
## 1: GENE_0356       S01     2   1.584963  FALSE
## 2: GENE_0233       S01    13   3.807355   TRUE
## 3: GENE_0232       S01    73   6.209453   TRUE
## 4: GENE_0160       S01    25   4.700440   TRUE
## 5: GENE_0049       S01     9   3.321928   TRUE
```

DATA.FRAME VERSION

```
##        gene sample_id count log2_count  high
## 1 GENE_0356       S01     2   1.584963 FALSE
## 2 GENE_0233       S01    13   3.807355  TRUE
## 3 GENE_0232       S01    73   6.209453  TRUE
## 4 GENE_0160       S01    25   4.700440  TRUE
## 5 GENE_0049       S01     9   3.321928  TRUE
```

TASK 3

The goal of this task is to increase the efficiency of dataset merging and repeated lookups. The task uses two input files: sample_metadata.csv, which provides sample annotations, and bulk_counts_long.csv, which contains the long-format count measurements. First, the sample_metadata table is indexed by sample_id using setkey(), and an equi-join is performed to merge it directly into the count dataset. Next, a secondary index on (gene, sample_id) is created in the counts table to accelerate filtering and lookup operations. To evaluate the performance improvement, a benchmark is run to compare subset queries before and after indexing.

DATA.TABLE VERSION

```
task3_dt <- function(counts_file = "bulk_counts_long.csv",
                     sample_metadata = "sample_metadata.csv") {

# Load needed libraries
library(data.table)
library(microbenchmark)

# Read the files
counts_dt3 <- fread(counts_file)
meta_dt3   <- fread(sample_metadata)

# Set key and join by sample_id
setkey(meta_dt3, sample_id)
merged_dt3 <- counts_dt3[meta_dt3, on = "sample_id", nomatch = 0L]


# Benchmark before and after index
gene_query   <- merged_dt3$gene[1]
sample_query <- merged_dt3$sample_id[1]

before_time <- microbenchmark(
no_index = merged_dt3[gene == gene_query & sample_id == sample_query],
times = 20L
)

# Add secondary index
setindexv(merged_dt3, c("gene", "sample_id"))
after_time <- microbenchmark(
    with_index = merged_dt3[gene == gene_query & sample_id == sample_query],
    times = 20L
)

# Combine benchmark results
benchmark3 <- rbind(
    data.table(expr = "no_index",  median_ms = median(before_time$time) / 1e6),
    data.table(expr = "with_index", median_ms = median(after_time$time) / 1e6)
  )
  benchmark3[, speedup := round(benchmark3$median_ms[1] / benchmark3$median_ms, 2)]

return(list(
```

```
merged_dt3 = merged_dt3,
benchmark3 = benchmark3
  ))
}

# SHOW RESULTS
res_dt3 <- task3_dt("bulk_counts_long.csv", "sample_metadata.csv")

head(res_dt3$merged_dt3, 5)
```

```
##         gene sample_id count condition  batch patient_id timepoint
##       <char>    <char> <int>    <char> <char>     <char>    <char>
## 1: GENE_0356       S01     2   treated batch1       P014        T0
## 2: GENE_0233       S01    13   treated batch1       P014        T0
## 3: GENE_0232       S01    73   treated batch1       P014        T0
## 4: GENE_0160       S01    25   treated batch1       P014        T0
## 5: GENE_0049       S01     9   treated batch1       P014        T0
```

```
head(res_dt3$benchmark3, 5)
```

```
##          expr median_ms speedup
##        <char>     <num>   <num>
## 1:   no_index 0.4042190       1
## 2: with_index 0.4030505       1
```

DATA.FRAME VERSION

```
task3_df <- function(counts_file = "bulk_counts_long.csv",
                     sample_metadata = "sample_metadata.csv") {

# Read data
counts_df3 <- read.csv(counts_file, stringsAsFactors = FALSE)
meta_df3   <- read.csv(sample_metadata, stringsAsFactors = FALSE)


# Merge using base R
merged_df3 <- merge(counts_df3, meta_df3, by = "sample_id", all = FALSE)

# Define query
gene_query   <- merged_df3$gene[1]
sample_query <- merged_df3$sample_id[1]


# Benchmark
benchmark_df3 <- microbenchmark(
  subset_lookup = merged_df3[merged_df3$gene == gene_query &
                             merged_df3$sample_id == sample_query, ],
  times = 100L
)
```

```r
# Subset
subset_df3 <- merged_df3[merged_df3$gene ==
                           gene_query & merged_df3$sample_id == sample_query, ]

return(list(
  merged_df3 = merged_df3,
  benchmark_df3 = benchmark_df3,
  subset_df3 = subset_df3))
}


# SHOW RESULTS

res_df3 <- task3_df("bulk_counts_long.csv", "sample_metadata.csv")

head(res_df3$merged_df3, 5)
```

```
##   sample_id      gene count condition  batch patient_id timepoint
## 1       S01 GENE_0356     2   treated batch1       P014        T0
## 2       S01 GENE_0233    13   treated batch1       P014        T0
## 3       S01 GENE_0232    73   treated batch1       P014        T0
## 4       S01 GENE_0160    25   treated batch1       P014        T0
## 5       S01 GENE_0049     9   treated batch1       P014        T0
```

```r
head(res_df3$benchmark_df3)
```

```
## Unit: microseconds
##           expr     min      lq     mean  median      uq     max neval
##  subset_lookup 123.656 126.526 138.8055 137.227 149.773 158.424     6
```

```r
head(res_df3$subset_df3, 5)
```

```
##   sample_id      gene count condition  batch patient_id timepoint
## 1       S01 GENE_0356     2   treated batch1       P014        T0
```

TASK 4

The goal of this task is to integrate sample-level metadata with the count matrix and derive summary statistics based on patient information. The analysis uses bulk_counts_long.csv (gene expression counts per sample) together with sample_metadata.csv (which includes patient and condition annotations). The two datasets are first joined, allowing each count value to be associated with the corresponding sample and patient. From this combined dataset, the total expression counts per patient are calculated. Furthermore, within each condition group, the top 10 genes with the highest mean expression values are identified.

DATA.TABLE VERSION

```r
task4_dt <- function(counts_file = "bulk_counts_long.csv",
                     sample_metadata = "sample_metadata.csv") {
# Load library
library(data.table)

# Read both files
counts_dt4 <- fread(counts_file)
meta_dt4   <- fread(sample_metadata)

# Annotate counts with metadata (join by sample_id)
counts_annotated <- merge(counts_dt4, meta_dt4, by = "sample_id", all.x = TRUE)

# Compute per-patient total counts (then sort by patient_id)
counts_patient_total <- counts_annotated[, .(total_count = sum(count)),
                                         by = patient_id]
setorder(counts_patient_total, patient_id)

# Find top 10 genes by average count within each condition
counts_top_genes <- counts_annotated[, .(avg_count = mean(count)),
                                     by = .(condition, gene)]

# Order descending and pick top 10 per condition
counts_top10_by_condition <- counts_top_genes[order(condition, -avg_count),
                                              .SD[1:10], by = condition]

  return(list(
    counts_annotated = counts_annotated,
    counts_patient_total = counts_patient_total,
    counts_top10_by_condition = counts_top10_by_condition
  ))
}

# SHOW RESULTS
res_dt4 <- task4_dt("bulk_counts_long.csv", "sample_metadata.csv")
head(res_dt4$counts_patient_total, 5)
```

```
##    patient_id total_count
##        <char>       <int>
## 1:       P002       11437
## 2:       P007       10859
## 3:       P008       22726
## 4:       P009       11338
## 5:       P011       11511
```

```r
head(res_dt4$counts_top10_by_condition, 5)
```

```
##    condition      gene avg_count
##       <char>    <char>     <num>
## 1:   control GENE_0272  179.3333
## 2:   control GENE_0091  169.8889
## 3:   control GENE_0425  160.7778
## 4:   control GENE_0377  160.3333
## 5:   control GENE_0262  147.3333
```

DATA.FRAME VERSION

```
task4_df <- function(counts_file = "bulk_counts_long.csv",
                     sample_metadata = "sample_metadata.csv") {

# Read data
counts_df4 <- read.csv(counts_file, stringsAsFactors = FALSE)
meta_df4   <- read.csv(sample_metadata, stringsAsFactors = FALSE)

# Annotate counts with metadata (prevent automatic sorting)
counts_annotated <- merge(counts_df4, meta_df4, by = "sample_id", all.x = TRUE,
                          sort = FALSE)

# Compute total counts per patient
counts_patient_totals <- aggregate(count ~ patient_id, data = counts_annotated,
                                   FUN = sum)
colnames(counts_patient_totals)[2] <- "total_count"

# Sort the data
counts_patient_totals <- counts_patient_totals[order(
                                    counts_patient_totals$patient_id), ]

# Find top 10 genes by average count within each condition
counts_avg_by_condition <- aggregate(count ~ condition + gene,
                                   data = counts_annotated, FUN = mean)
colnames(counts_avg_by_condition)[3] <- "avg_count"

# Order and extract top 10 per condition
counts_top10_genes <- do.call(rbind, lapply(split(counts_avg_by_condition, counts_avg_by_condition$condi
    x <- x[order(-x$avg_count), ]
    head(x, 10)
  }))

rownames(counts_top10_genes) <- NULL   # remove control.* rownames

  return(list(
    counts_annotated = counts_annotated,
    counts_patient_totals = counts_patient_totals,
    counts_top10_genes = counts_top10_genes
  ))
}

# SHOW RESULTS
res_df4 <- task4_df("bulk_counts_long.csv", "sample_metadata.csv")
head(res_df4$counts_patient_totals, 5)
```

```
##   patient_id total_count
## 1       P002       11437
## 2       P007       10859
## 3       P008       22726
## 4       P009       11338
## 5       P011       11511
```

```r
head(res_df4$counts_top10_genes, 5)
```

```
##   condition      gene avg_count
## 1   control GENE_0272  179.3333
## 2   control GENE_0091  169.8889
## 3   control GENE_0425  160.7778
## 4   control GENE_0377  160.3333
## 5   control GENE_0262  147.3333
```

TASK 5

The goal of this task is to classify clinical laboratory measurements according to reference intervals. The analysis uses two main input files: clinical_labs.csv, containing lab measurement values for each patient over time, and lab_reference_ranges.csv, which provides the corresponding lower and upper normal limits for each lab test. The reference ranges are treated as intervals, and each measurement is labeled as either "normal" or "out_of_range" using a non-equi join (i.e., value >= lower & value <= upper). After classification, the frequency of abnormal results is summarized both per patient and per lab test, enabling visualization of overall abnormality rates across the cohort.

DATA.TABLE VERSION

```r
task5_dt <- function(labs_file = "clinical_labs.csv",
                     ref_file  = "lab_reference_ranges.csv",
                     meta_file = "sample_metadata.csv") {

#Load library
  library(data.table)

# Read data
  labs_dt <- fread(labs_file)
  ref_dt  <- fread(ref_file)

# Remove duplicate M/F reference ranges
  ref_dt <- unique(ref_dt[, .(lab, lower, upper)])

# Merge by lab and classify using vectorised conditions
  merged <- merge(labs_dt, ref_dt, by = "lab", all.x = TRUE)

  merged[, status := fifelse(value >= lower & value <= upper,
                             "normal", "out_of_range")]
#Summaries
  abnormal_by_patient_dt5 <- merged[, .(
    total_labs = .N,
    abnormal_n = sum(status == "out_of_range"),
    abnormal_rate = mean(status == "out_of_range")
  ), by = patient_id]

  abnormal_by_lab_dt5 <- merged[, .(
    total_patients = .N,
```

```
    abnormal_n = sum(status == "out_of_range"),
    abnormal_rate = mean(status == "out_of_range")
  ), by = lab]

# Order
setorder(merged, patient_id, time_iso, lab)
setorder(abnormal_by_patient_dt5, -abnormal_rate)
setorder(abnormal_by_lab_dt5, -abnormal_rate)

return(list(
    classified_dt5 = merged[, .(patient_id, time_iso, lab, value, status)],
    abnormal_by_patient_dt5 = abnormal_by_patient_dt5,
    abnormal_by_lab_dt5 = abnormal_by_lab_dt5
  ))
}

# SHOW RESULTS
res5 <- task5_dt()
head(res5$classified_dt5)
```

```
##    patient_id            time_iso    lab    value       status
##        <char>              <POSc> <char>    <num>       <char>
## 1:       P002 2025-09-10 13:00:00    CRP 0.100000       normal
## 2:       P002 2025-09-10 13:00:00    WBC 6.332412       normal
## 3:       P002 2025-09-14 09:00:00    CRP 9.709428 out_of_range
## 4:       P002 2025-09-14 09:00:00    WBC 8.108361       normal
## 5:       P002 2025-09-16 09:00:00    CRP 6.359407 out_of_range
## 6:       P002 2025-09-16 09:00:00    WBC 5.464871       normal
```

```
head(res5$abnormal_by_patient_dt5)
```

```
##    patient_id total_labs abnormal_n abnormal_rate
##        <char>      <int>      <int>         <num>
## 1:       P053         20          9     0.4500000
## 2:       P011         14          6     0.4285714
## 3:       P036         14          6     0.4285714
## 4:       P007         16          6     0.3750000
## 5:       P018         14          5     0.3571429
## 6:       P044         20          7     0.3500000
```

```
head(res5$abnormal_by_lab_dt5)
```

```
## Key: <lab>
##       lab total_patients abnormal_n abnormal_rate
##    <char>          <int>      <int>         <num>
## 1:    CRP            160         76        0.4750
## 2:    WBC            160         18        0.1125
```

DATA.FRAME VERSION

```r
task5_df <- function(labs_file = "clinical_labs.csv",
                     ref_file  = "lab_reference_ranges.csv",
                     meta_file = "sample_metadata.csv") {

# Read data
  labs_df <- read.csv(labs_file, stringsAsFactors = FALSE)
  ref_df  <- read.csv(ref_file,  stringsAsFactors = FALSE)

# Remove duplicate M/F reference ranges
  ref_df <- unique(ref_df[, c("lab", "lower", "upper")])

# Merge and classify vectorised
  merged <- merge(labs_df, ref_df, by = "lab", all.x = TRUE)
  merged$status <- ifelse(merged$value >= merged$lower &
                            merged$value <= merged$upper,
                          "normal", "out_of_range")

# Summaries
abnormal_by_patient_df5 <- aggregate(status ~ patient_id, data = merged,
                                     FUN = function(x) mean(x == "out_of_range"))
 names(abnormal_by_patient_df5)[2] <- "abnormal_rate"

abnormal_by_patient_df5$total_labs <- as.numeric(table(merged$patient_id))
abnormal_by_patient_df5$abnormal_n <- round(
  abnormal_by_patient_df5$abnormal_rate * abnormal_by_patient_df5$total_labs, 0)

abnormal_by_lab_df5 <- aggregate(status ~ lab, data = merged,
                                 FUN = function(x) mean(x == "out_of_range"))
names(abnormal_by_lab_df5)[2] <- "abnormal_rate"
abnormal_by_lab_df5$total_patients <- as.numeric(table(merged$lab))
abnormal_by_lab_df5$abnormal_n <- round(
abnormal_by_lab_df5$abnormal_rate * abnormal_by_lab_df5$total_patients, 0)

# Order and return
  merged <- merged[order(merged$patient_id, merged$time_iso, merged$lab), ]
  abnormal_by_patient_df5 <- abnormal_by_patient_df5[
    order(-abnormal_by_patient_df5$abnormal_rate), ]
  abnormal_by_lab_df5 <- abnormal_by_lab_df5[
    order(-abnormal_by_lab_df5$abnormal_rate), ]

  return(list(
    classified_df5 = merged[, c("patient_id", "time_iso", "lab", "value", "status")],
    abnormal_by_patient_df5 = abnormal_by_patient_df5,
    abnormal_by_lab_df5 = abnormal_by_lab_df5
  ))
}

# SHOW RESULTS
res5_df <- task5_df()
head(res5_df$classified_df5)
```

```
##     patient_id            time_iso lab    value       status
## 1         P002 2025-09-10 13:00:00 CRP 0.100000       normal
```

```
## 161       P002 2025-09-10 13:00:00 WBC 6.332412       normal
## 3         P002 2025-09-14 09:00:00 CRP 9.709428 out_of_range
## 162       P002 2025-09-14 09:00:00 WBC 8.108361       normal
## 5         P002 2025-09-16 09:00:00 CRP 6.359407 out_of_range
## 163       P002 2025-09-16 09:00:00 WBC 5.464871       normal
```

```
head(res5_df$abnormal_by_patient_df5)
```

```
##    patient_id abnormal_rate total_labs abnormal_n
## 19       P053     0.4500000         20          9
## 5        P011     0.4285714         14          6
## 15       P036     0.4285714         14          6
## 2        P007     0.3750000         16          6
## 9        P018     0.3571429         14          5
## 17       P044     0.3500000         20          7
```

```
head(res5_df$abnormal_by_lab_df5)
```

```
##   lab abnormal_rate total_patients abnormal_n
## 1 CRP        0.4750            160         76
## 2 WBC        0.1125            160         18
```

TASK 6

The goal of this task is to classify clinical laboratory measurements according to reference intervals. The analysis uses two main input files: clinical_labs.csv, containing lab measurement values for each patient over time, and lab_reference_ranges.csv, which provides the corresponding lower and upper normal limits for each lab test. (Optionally, sample_metadata.csv can be used to incorporate sex-specific ranges.) The reference ranges are treated as intervals, and each measurement is labeled as either "normal" or "out_of_range" using a non-equi join (i.e., value >= lower & value <= upper). After classification, the frequency of abnormal results is summarized both per patient and per lab test, enabling visualization of overall abnormality rates across the cohort.

DATA.TABLE VERSION

```
##    patient_id    lab    value            lab_time          vital_time       HR
##        <char> <char>    <num>              <POSc>              <POSc>    <num>
## 1:       P002    CRP 0.100000 2025-09-10 13:00:00 2025-09-09 21:00:00 84.38194
## 2:       P002    WBC 6.332412 2025-09-10 13:00:00 2025-09-09 21:00:00 84.38194
## 3:       P002    CRP 9.709428 2025-09-14 09:00:00 2025-09-13 08:00:00 91.69549
## 4:       P002    WBC 8.108361 2025-09-14 09:00:00 2025-09-13 08:00:00 91.69549
## 5:       P002    CRP 6.359407 2025-09-16 09:00:00 2025-09-15 10:00:00 86.04024
##         SBP time_lag_min
##       <num>        <num>
## 1: 141.7723          960
## 2: 141.7723          960
## 3: 118.0909         1500
## 4: 118.0909         1500
## 5: 135.3029         1380
```

```
##    patient_id cor_CRP_HR cor_CRP_SBP
##        <char>      <num>       <num>
## 1:       P002  0.5295844  -0.8963853
## 2:       P007  0.3283777  -0.8886114
## 3:       P008 -0.3327580  -0.3181969
## 4:       P009  0.2591477   0.2829342
## 5:       P011  0.4510498   0.3182141
```

DATA.FRAME VERSION

```
##       patient_id lab    value            lab_time            time_lab       HR
## P002        P002 CRP 0.100000 2025-09-10 13:00:00 2025-09-09 21:00:00 84.38194
## P0021       P002 WBC 6.332412 2025-09-10 13:00:00 2025-09-09 21:00:00 84.38194
## P0022       P002 CRP 9.709428 2025-09-14 09:00:00 2025-09-13 08:00:00 91.69549
## P0023       P002 WBC 8.108361 2025-09-14 09:00:00 2025-09-13 08:00:00 91.69549
## P0024       P002 CRP 6.359407 2025-09-16 09:00:00 2025-09-15 10:00:00 86.04024
##             SBP time_lag_min
## P002  141.7723          960
## P0021 141.7723          960
## P0022 118.0909         1500
## P0023 118.0909         1500
## P0024 135.3029         1380
```

```
##      patient_id cor_CRP_HR cor_CRP_SBP
## P002       P002  0.5295844  -0.8963853
## P007       P007  0.3283777  -0.8886114
## P008       P008 -0.3327580  -0.3181969
## P009       P009  0.2591477   0.2829342
## P011       P011  0.4510498   0.3182141
```

TASK 7

The goal of this task is to filter and rank genomic regions (ATAC-seq peaks) within a specific chromosomal window. The analysis uses atac_peaks.bed.csv, which contains genomic peak coordinates and associated peak scores. The dataset is first restricted to peaks located on chromosome 2, with start coordinates between 2 Mb and 4 Mb. From this filtered subset, peaks are then ranked by score in descending order, and the top 50 highest-scoring peaks are returned as the final output.

DATA.TABLE VERSION

```r
task7_dt <- function(peaks_file = "atac_peaks.bed.csv") {
# Load library
library(data.table)

# Read the peaks file
peaks_dt <- fread(peaks_file)

# Subset peaks on chr2 in 2-4 Mb region
chr2_window <- peaks_dt[
```

```
  chr == "chr2" & start >= 2e6 & start <= 4e6]

# Order by descending score
setorder(chr2_window, -score)

# Select top 50 peaks (safe even if <50)
top50_peaks <- head(chr2_window, 50)

  return(list(
    chr2_window = chr2_window,
    top50_peaks = top50_peaks
  ))
}

# SHOW RESULTS
res_dt7 <- task7_dt("atac_peaks.bed.csv")
head(res_dt7$chr2_window, 5)
```

```
##       chr   start     end    peak_id score
##    <char>   <int>   <int>     <char> <int>
## 1:   chr2 3228165 3228913 peak_03666   989
## 2:   chr2 3810294 3811980 peak_03424   986
## 3:   chr2 3181717 3182848 peak_01057   979
## 4:   chr2 2104820 2105976 peak_03553   974
## 5:   chr2 2149706 2150080 peak_02715   974
```

```
head(res_dt7$top50_peaks, 5)
```

```
##       chr   start     end    peak_id score
##    <char>   <int>   <int>     <char> <int>
## 1:   chr2 3228165 3228913 peak_03666   989
## 2:   chr2 3810294 3811980 peak_03424   986
## 3:   chr2 3181717 3182848 peak_01057   979
## 4:   chr2 2104820 2105976 peak_03553   974
## 5:   chr2 2149706 2150080 peak_02715   974
```

DATA.FRAME VERSION

```
task7_df <- function(peaks_file = "atac_peaks.bed.csv") {
# Load library
library(readr)

# Read the peaks file
peaks_df <- read_csv(peaks_file)

# Show first few lines
head(peaks_df)

# Subset peaks on chr2 in 2-4 Mb region
chr2_window_df <- subset(
```

```
    peaks_df,
    chr == "chr2" & start >= 2e6 & start <= 4e6
  )

# Order by descending score
chr2_window_df <- chr2_window_df[order(-chr2_window_df$score), ]

# Select top 50 peaks
top50_peaks_df <- head(chr2_window_df, 50)

return(list(
    chr2_window_df = chr2_window_df,
    top50_peaks_df = top50_peaks_df
  ))
}


# SHOW RESULTS
res_df7 <- task7_df("atac_peaks.bed.csv")
```

```
## Rows: 5000 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): chr, peak_id
## dbl (3): start, end, score
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(res_df7$chr2_window_df, 5)
```

```
## # A tibble: 5 x 5
##   chr     start     end peak_id    score
##   <chr>   <dbl>   <dbl> <chr>      <dbl>
## 1 chr2  3228165 3228913 peak_03666   989
## 2 chr2  3810294 3811980 peak_03424   986
## 3 chr2  3181717 3182848 peak_01057   979
## 4 chr2  2104820 2105976 peak_03553   974
## 5 chr2  2149706 2150080 peak_02715   974
```

```
head(res_df7$top50_peaks_df, 5)
```

```
## # A tibble: 5 x 5
##   chr     start     end peak_id    score
##   <chr>   <dbl>   <dbl> <chr>      <dbl>
## 1 chr2  3228165 3228913 peak_03666   989
## 2 chr2  3810294 3811980 peak_03424   986
## 3 chr2  3181717 3182848 peak_01057   979
## 4 chr2  2104820 2105976 peak_03553   974
## 5 chr2  2149706 2150080 peak_02715   974
```

TASK 8

The goal of this task is to perform group-wise summary statistics across multiple columns. The analysis uses bulk_counts_long.csv (gene expression counts) joined with sample_metadata.csv (condition assignments). For each gene within each condition, several robust summary statistics are calculated, including the mean, median, and the first and third quartiles (Q1 and Q3). After computing these summaries, genes are filtered to retain only those for which the mean expression in the treated condition is at least twice the mean expression in the control condition. This identifies genes that demonstrate a strong differential expression pattern between conditions.

DATA.TABLE VERSION

```
task8_dt <- function(counts_file = "bulk_counts_long.csv",
                     meta_file   = "sample_metadata.csv") {

# Load library
library(data.table)

# Read the CSV files
counts_dt8 <- fread(counts_file)
meta_dt8   <- fread(meta_file)

# Merge counts with metadata by sample_id
merged_dt8 <- merge(counts_dt8, meta_dt8, by = "sample_id", sort = FALSE)

# Compute per-condition robust summary stats for each gene (mean, median, Q1, Q3)
statsum_dt8 <- merged_dt8[, .(
mean_count   = mean(count, na.rm = TRUE),
median_count = median(count, na.rm = TRUE),
    Q1           = quantile(count, 0.25, na.rm = TRUE),
    Q3           = quantile(count, 0.75, na.rm = TRUE)
  ), by = .(gene, condition)]

# View summary
print(head(statsum_dt8))

# Reshape to wide format (one row per gene, columns for each condition)
wide_dt8 <- dcast(statsum_dt8, gene ~ condition, value.var = "mean_count")

# Filter by keeping only genes where treated mean  2 × control mean
filtered_dt8 <- wide_dt8[
  !is.na(.SD$treated) &
    !is.na(.SD$control) &
    .SD$treated >= 2 * .SD$control
]

return(list(
  statsum_dt8  = statsum_dt8,
  wide_dt8     = wide_dt8,
  filtered_dt8 = filtered_dt8
  ))
}
```

```
# SHOW RESULTS
counts_dt8_result <- task8_dt()
```

```
##         gene condition mean_count median_count    Q1    Q3
##       <char>    <char>      <num>        <int> <num> <num>
## 1: GENE_0356   treated   3.866667            3   2.0   5.5
## 2: GENE_0233   treated   9.533333           10   6.0  11.5
## 3: GENE_0232   treated  71.200000           69  61.5  76.5
## 4: GENE_0160   treated  22.066667           23  18.0  25.5
## 5: GENE_0049   treated   7.466667            7   6.0   9.0
## 6: GENE_0245   treated  32.133333           33  24.5  37.5
```

```
head(counts_dt8_result$statsum_dt8, 5)
```

```
##         gene condition mean_count median_count    Q1    Q3
##       <char>    <char>      <num>        <int> <num> <num>
## 1: GENE_0356   treated   3.866667            3   2.0   5.5
## 2: GENE_0233   treated   9.533333           10   6.0  11.5
## 3: GENE_0232   treated  71.200000           69  61.5  76.5
## 4: GENE_0160   treated  22.066667           23  18.0  25.5
## 5: GENE_0049   treated   7.466667            7   6.0   9.0
```

```
head(counts_dt8_result$wide_dt8, 5)
```

```
## Key: <gene>
##         gene   control   treated
##       <char>     <num>     <num>
## 1: GENE_0000  6.111111  7.066667
## 2: GENE_0001  2.666667  2.533333
## 3: GENE_0002  4.000000  3.800000
## 4: GENE_0003 12.777778 20.266667
## 5: GENE_0004  8.444444  5.933333
```

```
head(counts_dt8_result$filtered_dt8, 5)
```

```
## Key: <gene>
## Empty data.table (0 rows and 3 cols): gene,control,treated
```

DATA.FRAME VERSION

```
task8_df <- function(counts_file = "bulk_counts_long.csv",
                     meta_file   = "sample_metadata.csv") {

# Load libraries
library(dplyr)
library(tidyr)
library(readr)
```

```r
# Read CSV files
counts_df8 <- read_csv(counts_file)
meta_df8   <- read_csv(meta_file)

# Merge counts with metadata by sample_id
merged_df8 <- merge(counts_df8, meta_df8, by = "sample_id", sort = FALSE)

# Compute per-condition summary stats for each gene
statsum_df8 <- merged_df8 %>%
group_by(gene, condition) %>%
    summarise(
      mean_count   = mean(count, na.rm = TRUE),
      median_count = median(count, na.rm = TRUE),
      Q1           = quantile(count, 0.25, na.rm = TRUE),
      Q3           = quantile(count, 0.75, na.rm = TRUE),
      .groups = "drop"
    )

# View summary
print(head(statsum_df8))

# Reshape from long → wide so we can compare control vs treated
wide_df8 <- statsum_df8 %>%
  select(gene, condition, mean_count) %>%
  pivot_wider(names_from = condition, values_from = mean_count)

# Filter genes where treated mean  2 × control mean
filtered_df8 <- wide_df8 %>%
filter(!is.na(treated) & !is.na(control) & treated >= 2 * control)

return(list(
statsum_df8  = statsum_df8,
  wide_df8     = wide_df8,
   filtered_df8 = filtered_df8
  ))
}

# SHOW RESULTS
counts_df8_result <- task8_df()
```

```
## # A tibble: 6 x 6
##   gene      condition mean_count median_count    Q1    Q3
##   <chr>     <chr>          <dbl>        <dbl> <dbl> <dbl>
## 1 GENE_0000 control         6.11            6     5   9
## 2 GENE_0000 treated         7.07            6     5   7.5
## 3 GENE_0001 control         2.67            3     2   3
## 4 GENE_0001 treated         2.53            2     1   3.5
## 5 GENE_0002 control         4               4     3   5
## 6 GENE_0002 treated         3.8             3     2   5
```

```r
head(counts_df8_result$statsum_df8, 5)
```

```
## # A tibble: 5 x 6
```

```
##   gene      condition mean_count median_count    Q1    Q3
##   <chr>     <chr>          <dbl>        <dbl> <dbl> <dbl>
## 1 GENE_0000 control         6.11            6     5   9
## 2 GENE_0000 treated         7.07            6     5   7.5
## 3 GENE_0001 control         2.67            3     2   3
## 4 GENE_0001 treated         2.53            2     1   3.5
## 5 GENE_0002 control         4               4     3   5
```

```
head(counts_df8_result$wide_df8, 5)
```

```
## # A tibble: 5 x 3
##   gene      control treated
##   <chr>       <dbl>   <dbl>
## 1 GENE_0000    6.11    7.07
## 2 GENE_0001    2.67    2.53
## 3 GENE_0002    4       3.8
## 4 GENE_0003   12.8    20.3
## 5 GENE_0004    8.44    5.93
```

```
head(counts_df8_result$filtered_df8, 5)
```

```
## # A tibble: 5 x 3
##   gene      control treated
##   <chr>       <dbl>   <dbl>
## 1 GENE_0020  28.4     93.7
## 2 GENE_0038   2.44     5.07
## 3 GENE_0040   3.67    12.9
## 4 GENE_0081   0.444    1.07
## 5 GENE_0088   8.89    29.4
```

TASK 9

The goal of this task is to perform group-wise summary statistics across multiple columns. The analysis uses bulk_counts_long.csv (gene expression counts) joined with sample_metadata.csv (condition assignments). For each gene within each condition, several robust summary statistics are calculated, including the mean, median, and the first and third quartiles (Q1 and Q3). After computing these summaries, genes are filtered to retain only those for which the mean expression in the treated condition is at least twice the mean expression in the control condition. This identifies genes that demonstrate a strong differential expression pattern between conditions.

DATA.TABLE VERSION

```
task9_dt <- function(wide_file = "bulk_counts_wide.csv") {

# Load library
library(data.table)

# Read CSV file
counts_wide_dt <- fread(wide_file)   # columns: gene, S01, S02, …
```

```r
str(counts_wide_dt)

# Identify numeric columns (sample count columns)
numeric_columns <- setdiff(names(counts_wide_dt), "gene")
numeric_columns  # should list S01, S02, …

# Convert from wide to long format
counts_long_dt <- melt(
  counts_wide_dt,
  id.vars = "gene",
  measure.vars = numeric_columns,
  variable.name = "sample_id",
  value.name = "count"
  )

setDT(counts_long_dt)

# Compute per-sample totals
counts_long_dt[, sample_total := sum(count, na.rm = TRUE), by = sample_id]

# Assign condition labels (example rule)
counts_long_dt[, condition := ifelse(
  grepl("treat", sample_id, ignore.case = TRUE),
    "treated", "control"
  )]

# Compute mean count per gene × condition
counts_mean_dt <- counts_long_dt[
    , .(mean_count = mean(count, na.rm = TRUE)),
    by = .(gene, condition)
  ][order(gene, condition)]

# Convert back to wide format (gene × condition)
counts_summary_dt <- dcast(
  counts_mean_dt,
  gene ~ condition,
  value.var = "mean_count"
  )

# Set outputs as data.table
setDT(counts_long_dt)
setDT(counts_mean_dt)
setDT(counts_summary_dt)


return(list(
  counts_long_dt    = counts_long_dt,
  counts_mean_dt    = counts_mean_dt,
  counts_summary_dt = counts_summary_dt
  ))
}
```

```r
# SHOW RESULTS
counts_dt9_result <- task9_dt("bulk_counts_wide.csv")
```

```
## Classes 'data.table' and 'data.frame':   500 obs. of  25 variables:
##  $ gene: chr  "GENE_0356" "GENE_0233" "GENE_0232" "GENE_0160" ...
##  $ S01 : int  2 13 73 25 9 20 39 48 10 10 ...
##  $ S02 : int  3 8 70 24 4 29 38 18 9 8 ...
##  $ S03 : int  3 10 68 18 7 52 35 47 16 13 ...
##  $ S04 : int  6 6 56 16 5 29 44 24 20 6 ...
##  $ S05 : int  5 7 89 32 10 33 57 34 18 8 ...
##  $ S06 : int  5 13 80 26 10 22 44 30 17 11 ...
##  $ S07 : int  6 13 61 19 6 38 44 19 16 21 ...
##  $ S08 : int  2 10 71 15 1 15 47 26 14 11 ...
##  $ S09 : int  1 13 56 38 10 22 23 22 20 7 ...
##  $ S10 : int  4 5 69 18 12 44 70 25 16 8 ...
##  $ S11 : int  4 11 61 23 6 37 54 37 8 7 ...
##  $ S12 : int  8 16 62 25 9 30 51 32 12 8 ...
##  $ S13 : int  4 11 79 18 3 32 55 16 9 2 ...
##  $ S14 : int  9 6 69 28 7 37 45 18 5 5 ...
##  $ S15 : int  2 6 48 26 9 45 47 29 12 9 ...
##  $ S16 : int  3 11 80 32 7 42 34 26 14 13 ...
##  $ S17 : int  5 6 68 23 5 36 48 25 9 15 ...
##  $ S18 : int  1 6 77 22 11 19 37 24 5 9 ...
##  $ S19 : int  6 11 82 32 6 15 70 45 8 10 ...
##  $ S20 : int  4 16 57 30 5 21 52 39 25 7 ...
##  $ S21 : int  5 16 51 22 5 22 65 25 8 7 ...
##  $ S22 : int  1 10 76 18 11 31 44 37 11 3 ...
##  $ S23 : int  1 12 109 9 9 18 60 36 15 9 ...
##  $ S24 : int  2 7 70 23 4 34 35 25 11 3 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
head(counts_dt9_result$counts_long_dt, 5)
```

```
##         gene sample_id count sample_total condition
##       <char>    <fctr> <int>        <int>    <char>
## 1: GENE_0356       S01     2        11159   control
## 2: GENE_0233       S01    13        11159   control
## 3: GENE_0232       S01    73        11159   control
## 4: GENE_0160       S01    25        11159   control
## 5: GENE_0049       S01     9        11159   control
```

```r
head(counts_dt9_result$counts_mean_dt, 5)
```

```
##         gene condition mean_count
##       <char>    <char>      <num>
## 1: GENE_0000   control   6.708333
## 2: GENE_0001   control   2.583333
## 3: GENE_0002   control   3.875000
## 4: GENE_0003   control  17.458333
## 5: GENE_0004   control   6.875000
```

```r
head(counts_dt9_result$counts_summary_dt, 5)
```

```
## Key: <gene>
##         gene   control
##       <char>     <num>
## 1: GENE_0000  6.708333
## 2: GENE_0001  2.583333
## 3: GENE_0002  3.875000
## 4: GENE_0003 17.458333
## 5: GENE_0004  6.875000
```

DATA.FRAME VERSION

```r
task9_df <- function(wide_file = "bulk_counts_wide.csv") {

# Load library
library(reshape2)

# Read CSV file
counts_wide_df <- read.csv(wide_file, stringsAsFactors = FALSE)
head(counts_wide_df)
str(counts_wide_df)

# Identify numeric columns (sample count columns)
numeric_columns <- setdiff(names(counts_wide_df), "gene")
numeric_columns

# Convert from wide to long format
counts_long_df <- melt(
counts_wide_df,
id.vars = "gene",
measure.vars = numeric_columns,
variable.name = "sample_id",
value.name = "count"
  )

# Compute per-sample totals
counts_long_df$sample_total <- ave(
counts_long_df$count,
counts_long_df$sample_id,
FUN = function(x) sum(x, na.rm = TRUE)
  )


# Assign condition labels (example rule)
counts_long_df$condition <- ifelse(
  grepl("treat", counts_long_df$sample_id, ignore.case = TRUE),
  "treated", "control"
  )
```

```r
# Compute mean count per gene × condition
counts_mean_df <- aggregate(
  count ~ gene + condition,
  data = counts_long_df,
  FUN = function(x) mean(x, na.rm = TRUE)
)

names(counts_mean_df)[3] <- "mean_count"

counts_mean_df <- counts_mean_df[order(counts_mean_df$gene,
                                       counts_mean_df$condition), ]

# Convert back to wide format (gene × condition)
counts_summary_df <- reshape(
  counts_mean_df,
  timevar = "condition",
  idvar   = "gene",
  direction = "wide"
)

#Modify names
names(counts_summary_df) <- gsub("^mean_count\\.", "",
                                 names(counts_summary_df))

#Sort
col_order <- c("gene", setdiff(sort(names(counts_summary_df)), "gene"))
counts_summary_df <- counts_summary_df[, col_order]


return(list(
  counts_long_df    = counts_long_df,
  counts_mean_df    = counts_mean_df,
  counts_summary_df = counts_summary_df
  ))
}


# SHOW RESULTS
counts_df9_result <- task9_df("bulk_counts_wide.csv")
```

```
## 'data.frame':    500 obs. of  25 variables:
##  $ gene: chr  "GENE_0356" "GENE_0233" "GENE_0232" "GENE_0160" ...
##  $ S01 : int  2 13 73 25 9 20 39 48 10 10 ...
##  $ S02 : int  3 8 70 24 4 29 38 18 9 8 ...
##  $ S03 : int  3 10 68 18 7 52 35 47 16 13 ...
##  $ S04 : int  6 6 56 16 5 29 44 24 20 6 ...
##  $ S05 : int  5 7 89 32 10 33 57 34 18 8 ...
##  $ S06 : int  5 13 80 26 10 22 44 30 17 11 ...
##  $ S07 : int  6 13 61 19 6 38 44 19 16 21 ...
##  $ S08 : int  2 10 71 15 1 15 47 26 14 11 ...
##  $ S09 : int  1 13 56 38 10 22 23 22 20 7 ...
##  $ S10 : int  4 5 69 18 12 44 70 25 16 8 ...
##  $ S11 : int  4 11 61 23 6 37 54 37 8 7 ...
```

```
##  $ S12 : int  8 16 62 25 9 30 51 32 12 8 ...
##  $ S13 : int  4 11 79 18 3 32 55 16 9 2 ...
##  $ S14 : int  9 6 69 28 7 37 45 18 5 5 ...
##  $ S15 : int  2 6 48 26 9 45 47 29 12 9 ...
##  $ S16 : int  3 11 80 32 7 42 34 26 14 13 ...
##  $ S17 : int  5 6 68 23 5 36 48 25 9 15 ...
##  $ S18 : int  1 6 77 22 11 19 37 24 5 9 ...
##  $ S19 : int  6 11 82 32 6 15 70 45 8 10 ...
##  $ S20 : int  4 16 57 30 5 21 52 39 25 7 ...
##  $ S21 : int  5 16 51 22 5 22 65 25 8 7 ...
##  $ S22 : int  1 10 76 18 11 31 44 37 11 3 ...
##  $ S23 : int  1 12 109 9 9 18 60 36 15 9 ...
##  $ S24 : int  2 7 70 23 4 34 35 25 11 3 ...
```

```
head(counts_df9_result$counts_long_df, 5)
```

```
##        gene sample_id count sample_total condition
## 1 GENE_0356       S01     2        11159   control
## 2 GENE_0233       S01    13        11159   control
## 3 GENE_0232       S01    73        11159   control
## 4 GENE_0160       S01    25        11159   control
## 5 GENE_0049       S01     9        11159   control
```

```
head(counts_df9_result$counts_mean_df, 5)
```

```
##        gene condition mean_count
## 1 GENE_0000   control   6.708333
## 2 GENE_0001   control   2.583333
## 3 GENE_0002   control   3.875000
## 4 GENE_0003   control  17.458333
## 5 GENE_0004   control   6.875000
```

```
head(counts_df9_result$counts_summary_df, 5)
```

```
##        gene   control
## 1 GENE_0000  6.708333
## 2 GENE_0001  2.583333
## 3 GENE_0002  3.875000
## 4 GENE_0003 17.458333
## 5 GENE_0004  6.875000
```

TASK 10

The objective of this task is to map ATAC-seq peaks to gene bodies in order to estimate regulatory signal associated with each gene. The analysis uses two genomic interval datasets: atac_peaks.bed.csv (peak regions with scores) and gene_annotation.bed.csv (gene coordinates). Both datasets are keyed by their genomic coordinates (chr, start, end) to enable efficient interval-based operations. ATAC peaks that overlap gene bodies are identified, and for each gene, two types of summaries are generated: The number of peaks overlapping the gene, and The total number of overlapping base pairs, calculated by determining the overlap length for each peak–gene pair and summing across all peaks for that gene. Finally, the genes are ranked by total overlap length, and the top 20 genes with the highest cumulative peak coverage are returned.

DATA.TABLE VERSION

```
task10_dt <- function(peaks_file = "atac_peaks.bed.csv",
                      genes_file = "gene_annotation.bed.csv") {

library(data.table)

#Read data
peaks <- fread(peaks_file)
genes <- fread(genes_file)

#Standardize column names
setnames(peaks, 1:3, c("chr", "start", "end"))
setnames(genes, 1:4, c("chr", "gene_start", "gene_end", "gene"))

# Set keys
setkey(peaks, chr, start, end)
setkey(genes, chr, gene_start, gene_end)

#Intersect peaks with gene bodies
overlaps <- foverlaps(peaks, genes,
                        by.x = c("chr", "start", "end"),
                        type = "any", nomatch = 0L)

#Compute overlap length (bp)
overlaps[, overlap_bp := pmin(end, gene_end) - pmax(start, gene_start)]
overlaps <- overlaps[overlap_bp > 0]

#Count peaks and sum overlap per gene
peaks_per_gene <- overlaps[, .(
    n_peaks = .N,
    total_overlap_bp = sum(overlap_bp)
  ), by = gene]

#Top 20 genes by total overlap
top20 <- peaks_per_gene[order(-total_overlap_bp)][1:20]

return(list(
    overlaps_dt10 = overlaps,
    peaks_per_gene_dt10 = peaks_per_gene,
    top20_dt10 = top20
  ))
}

# SHOW RESULTS
res_dt10 <- task10_dt("atac_peaks.bed.csv", "gene_annotation.bed.csv")
head(res_dt10$overlaps_dt10)
```

```
## Key: <chr, start, end>
##       chr gene_start gene_end      gene   start     end    peak_id score
##    <char>      <int>    <int>    <char>   <int>   <int>     <char> <int>
## 1:   chr1    1064323  1084034 GENE_0356 1067760 1068860 peak_02815   425
## 2:   chr1    1317537  1346385 GENE_0233 1325076 1325461 peak_04736   675
```

```
## 3:   chr1    1317537  1346385 GENE_0233 1337256 1338396 peak_00981   143
## 4:   chr1    1327359  1353543 GENE_0232 1337256 1338396 peak_00981   143
## 5:   chr1    1317537  1346385 GENE_0233 1341992 1343938 peak_04765   174
## 6:   chr1    1327359  1353543 GENE_0232 1341992 1343938 peak_04765   174
##    overlap_bp
##         <int>
## 1:       1100
## 2:        385
## 3:       1140
## 4:       1140
## 5:       1946
## 6:       1946
```

```
head(res_dt10$peaks_per_gene_dt10)
```

```
##         gene n_peaks total_overlap_bp
##       <char>   <int>            <int>
## 1: GENE_0356       1             1100
## 2: GENE_0233       4             4867
## 3: GENE_0232       4             4976
## 4: GENE_0049       1              954
## 5: GENE_0245       2             1399
## 6: GENE_0130       1              509
```

```
res_dt10$top20_dt10
```

```
##          gene n_peaks total_overlap_bp
##        <char>   <int>            <int>
##  1: GENE_0437      12            13421
##  2: GENE_0431       8            12228
##  3: GENE_0346       9            11523
##  4: GENE_0105       8            10814
##  5: GENE_0256       9            10808
##  6: GENE_0349       7             9592
##  7: GENE_0189       8             9445
##  8: GENE_0388       7             9411
##  9: GENE_0305       8             8766
## 10: GENE_0162       8             8571
## 11: GENE_0094       8             8480
## 12: GENE_0291       7             8379
## 13: GENE_0353       8             8360
## 14: GENE_0186       8             8348
## 15: GENE_0047       6             8250
## 16: GENE_0236       8             8192
## 17: GENE_0133       6             8071
## 18: GENE_0157       8             7895
## 19: GENE_0141       7             7791
## 20: GENE_0128       5             7614
##          gene n_peaks total_overlap_bp
```

DATA.FRAME VERSION

```
task10_df <- function(peaks_file = "atac_peaks.bed.csv",
                      genes_file = "gene_annotation.bed.csv") {

# Read data
peaks <- read.csv(peaks_file)
genes <- read.csv(genes_file)

names(peaks)[1:3] <- c("chr", "start", "end")
names(genes)[1:4] <- c("chr", "gene_start", "gene_end", "gene")

#Container for overlaps
overlap_list <- list()

#Loop per chromosome
  for (ch in intersect(unique(peaks$chr), unique(genes$chr))) {
    p_sub <- subset(peaks, chr == ch)
    g_sub <- subset(genes, chr == ch)

#For each gene, find overlapping peaks
for (i in seq_len(nrow(g_sub))) {
      g <- g_sub[i, ]
      hits <- which(p_sub$end > g$gene_start & p_sub$start < g$gene_end)

if (length(hits) > 0) {
        tmp <- p_sub[hits, ]
        tmp$gene <- g$gene
        tmp$overlap_bp <- pmin(tmp$end, g$gene_end) - pmax(tmp$start, g$gene_start)
        tmp <- tmp[tmp$overlap_bp > 0, ]
        overlap_list[[length(overlap_list) + 1]] <- tmp
      }
    }
  }

# Combine overlaps
overlaps <- do.call(rbind, overlap_list)

# Order
overlaps <- overlaps[order(overlaps$chr, overlaps$start, overlaps$end), ]


#Summarize per gene
peaks_per_gene <- do.call(rbind,
                          lapply(split(overlaps, overlaps$gene), function(x)
                            data.frame(
                              gene = unique(x$gene),
                              n_peaks = nrow(x),
                              total_overlap_bp = sum(x$overlap_bp)
                            ))
  )

# Top 20 genes
```

```
top20 <- head(peaks_per_gene[order(-peaks_per_gene$total_overlap_bp), ], 20)

return(list(
    overlaps_df10 = overlaps,
    peaks_per_gene_df10 = peaks_per_gene,
    top20_df10 = top20
  ))
}


# SHOW RESULTS
res_df10 <- task10_df("atac_peaks.bed.csv", "gene_annotation.bed.csv")
head(res_df10$overlaps_df10)
```

```
##      chr   start     end    peak_id score      gene overlap_bp
## 66  chr1 1067760 1068860 peak_02815   425 GENE_0356       1100
## 94  chr1 1325076 1325461 peak_04736   675 GENE_0233        385
## 95  chr1 1337256 1338396 peak_00981   143 GENE_0233       1140
## 951 chr1 1337256 1338396 peak_00981   143 GENE_0232       1140
## 96  chr1 1341992 1343938 peak_04765   174 GENE_0233       1946
## 961 chr1 1341992 1343938 peak_04765   174 GENE_0232       1946
```

```
head(res_df10$peaks_per_gene_df10)
```

```
##                gene n_peaks total_overlap_bp
## GENE_0000 GENE_0000       2             1373
## GENE_0002 GENE_0002       2             3007
## GENE_0003 GENE_0003       1             1470
## GENE_0004 GENE_0004       1              136
## GENE_0005 GENE_0005       3             3284
## GENE_0007 GENE_0007       1              610
```

```
res_df10$top20_df10
```

```
##                gene n_peaks total_overlap_bp
## GENE_0437 GENE_0437      12            13421
## GENE_0431 GENE_0431       8            12228
## GENE_0346 GENE_0346       9            11523
## GENE_0105 GENE_0105       8            10814
## GENE_0256 GENE_0256       9            10808
## GENE_0349 GENE_0349       7             9592
## GENE_0189 GENE_0189       8             9445
## GENE_0388 GENE_0388       7             9411
## GENE_0305 GENE_0305       8             8766
## GENE_0162 GENE_0162       8             8571
## GENE_0094 GENE_0094       8             8480
## GENE_0291 GENE_0291       7             8379
## GENE_0353 GENE_0353       8             8360
## GENE_0186 GENE_0186       8             8348
## GENE_0047 GENE_0047       6             8250
## GENE_0236 GENE_0236       8             8192
```

```
## GENE_0133 GENE_0133       6            8071
## GENE_0157 GENE_0157       8            7895
## GENE_0141 GENE_0141       7            7791
## GENE_0128 GENE_0128       5            7614
```

TASK 11

The goal of this task is to map genomic variants to genes by overlapping each variant position with gene coordinates. Variants are converted to 1-bp intervals and matched to genes using interval overlap. The number of HIGH-impact variants is then summarized per gene and per sample, and the list of genes that contain HIGH-impact variants in any sample is reported.

DATA.TABLE VERSION

```
task11_dt <- function(variants_file = "variants.csv",
                      genes_file    = "gene_annotation.bed.csv") {

#Load library
library(data.table)

# Read data
  variants <- fread(variants_file, colClasses = "character")
  genes    <- fread(genes_file,    colClasses = "character")

# Convert numeric columns
variants[, pos := as.integer(pos)]
genes[, c("start","end") := .(as.integer(start), as.integer(end))]

#Create overlaps
overlaps_list <- lapply(1:nrow(variants), function(i) {
    v <- variants[i]
    subset <- genes[chr == v$chr & start <= v$pos & end >= v$pos]
    if (nrow(subset) > 0) {
      data.table(chr = v$chr,
                 gene = subset$gene,
                 sample_id = v$sample_id,
                 impact = v$impact)
    } else NULL
  })

overlaps <- rbindlist(overlaps_list, use.names = TRUE, fill = TRUE)

# Count HIGH-impact variants
counts <- overlaps[impact == "HIGH", .(n_high = .N), by = .(gene, sample_id)]
setorder(counts, gene, sample_id)

# Genes with HIGH-impact variants in all samples
n_samples <- uniqueN(variants$sample_id)
genes_all <- counts[, .N, by = gene][N == n_samples, gene]
```

```
return(list(
    overlaps_dt11      = overlaps,
    counts_per_gene_dt = counts,
    genes_high_all_dt  = genes_all
  ))
}
#SHOW RESULTS
res_dt11 <- task11_dt("variants.csv", "gene_annotation.bed.csv")
head(res_dt11$overlaps_dt11, 5)
```

```
##       chr      gene sample_id   impact
##    <char>    <char>    <char>   <char>
## 1:   chr1 GENE_0356       S12 MODERATE
## 2:   chr1 GENE_0049       S19     HIGH
## 3:   chr1 GENE_0049       S16      LOW
## 4:   chr1 GENE_0049       S15 MODERATE
## 5:   chr1 GENE_0049       S20      LOW
```

```
head(res_dt11$counts_per_gene_dt, 5)
```

```
##         gene sample_id n_high
##       <char>    <char>  <int>
## 1: GENE_0008       S07      1
## 2: GENE_0009       S09      1
## 3: GENE_0036       S14      1
## 4: GENE_0049       S19      1
## 5: GENE_0058       S07      1
```

```
res_dt11$genes_high_all_dt
```

```
## character(0)
```

DATA.FRAME VERSION

```
task11_df <- function(variants_file = "variants.csv",
                      genes_file    = "gene_annotation.bed.csv") {

#Read data
variants <- read.csv(variants_file, stringsAsFactors = FALSE)
genes    <- read.csv(genes_file, stringsAsFactors = FALSE)

variants$chr <- as.character(variants$chr)
variants$pos <- as.integer(variants$pos)
genes$chr    <- as.character(genes$chr)
genes$start  <- as.integer(genes$start)
genes$end    <- as.integer(genes$end)
genes$gene   <- as.character(genes$gene)

# Create overlaps
```

```r
overlaps <- data.frame(chr=character(), gene=character(),
                       sample_id=character(), impact=character(),
                       stringsAsFactors = FALSE)

for (i in 1:nrow(variants)) {
    v <- variants[i, ]
    matches <- subset(genes, chr == v$chr &
                      start <= v$pos &
                      end >= v$pos)
if (nrow(matches) > 0) {
    new_rows <- data.frame(chr = v$chr,
                           gene = matches$gene,
                           sample_id = v$sample_id,
                           impact = v$impact,
                           stringsAsFactors = FALSE)
      overlaps <- rbind(overlaps, new_rows)
    }
  }

# Count HIGH-impact variants
high <- subset(overlaps, impact == "HIGH")
if (nrow(high) == 0) {
    counts <- data.frame(gene=character(), sample_id=character(), n_high=integer())
  } else {
    counts <- aggregate(
      list(n_high = rep(1, nrow(high))),
      by = list(gene = high$gene, sample_id = high$sample_id),
      FUN = sum
    )
    counts <- counts[order(counts$gene, counts$sample_id), ]
  }

# Genes with HIGH-impact variants in all samples
n_samples <- length(unique(variants$sample_id))
  if (nrow(counts) == 0) {
    genes_all <- character(0)
  } else {
    gene_counts <- aggregate(sample_id ~ gene, data = counts,
                             FUN = function(x) length(unique(x)))
    genes_all <- sort(gene_counts$gene[gene_counts$sample_id == n_samples])
  }


  return(list(
    overlaps_df11      = overlaps,
    counts_per_gene_df = counts,
    genes_high_all_df  = genes_all
  ))
}

#SHOW RESULTS
res_df11 <- task11_df("variants.csv", "gene_annotation.bed.csv")
head(res_df11$overlaps_df11, 5)
```

```
##    chr      gene sample_id   impact
## 1 chr1 GENE_0356       S12 MODERATE
## 2 chr1 GENE_0049       S19     HIGH
## 3 chr1 GENE_0049       S16      LOW
## 4 chr1 GENE_0049       S15 MODERATE
## 5 chr1 GENE_0049       S20      LOW
```

```
head(res_df11$counts_per_gene_df, 5)
```

```
##         gene sample_id n_high
## 7  GENE_0008       S07      1
## 10 GENE_0009       S09      1
## 19 GENE_0036       S14      1
## 30 GENE_0049       S19      1
## 8  GENE_0058       S07      1
```

```
res_df11$genes_high_all_df
```

```
## character(0)
```

TASK 12

The goal of this task is to combine two study cohorts into a single dataset, ensuring that sample annotations are aligned correctly. The input files cohortA_samples.csv and cohortB_samples.csv are first merged using rbindlist(..., use.names = TRUE, fill = TRUE) to guarantee that columns match even if the two cohorts contain slightly different metadata fields. The combined dataset is then ordered by cohort, condition, and sample_id for consistency. Next, the merged cohort metadata is joined with bulk_counts_long.csv to associate each count measurement with its cohort. From this integrated dataset, the top 100 most variable genes are identified, and mean expression values are computed per cohort and per condition to allow comparison of expression patterns across study groups.

DATA.TABLE VERSION

```
task12_dt <- function(cohortA_file = "cohortA_samples.csv",
                      cohortB_file = "cohortB_samples.csv",
                      counts_file  = "bulk_counts_long.csv") {

# Load library
  library(data.table)

# Read cohort files
A <- fread(cohortA_file)
B <- fread(cohortB_file)

#  Combine cohorts
combined <- rbindlist(list(A, B), use.names = TRUE, fill = TRUE)

# Order by cohort, condition, sample_id
order_cols <- intersect(c("cohort", "condition", "sample_id"), names(combined))
```

```r
if (length(order_cols) > 0) setorderv(combined, order_cols)

# Read counts and join by sample_id
counts <- fread(counts_file)
merged <- merge(counts, combined, by = "sample_id", allow.cartesian = TRUE)

# Compute variance per gene (sorted descending)
var_dt <- merged[, .(variance = var(count, na.rm = TRUE)), by = gene]
var_dt <- var_dt[order(-variance, gene)]
top_genes <- var_dt$gene[1:100]

# Compute per-cohort/per-condition mean counts
summary_dt <- merged[gene %in% top_genes,
                     .(mean_count = mean(count, na.rm = TRUE)),
                     by = .(gene, cohort, condition)]
setorderv(summary_dt, c("cohort", "condition", "gene"))

# Column alignment summary (silent)
alignment_summary <- data.table(
  Status = c("In both", "Only in A", "Only in B"),
  Count  = c(length(intersect(names(A), names(B))),
             length(setdiff(names(A), names(B))),
             length(setdiff(names(B), names(A))))
  )

# Return all results
return(list(
    combined_samples = combined,
    merged_counts    = merged,
    top_genes_var    = var_dt,
    summary_counts   = summary_dt,
    column_check     = alignment_summary
  ))
}

# SHOW RESULTS
res_dt12 <- task12_dt("cohortA_samples.csv", "cohortB_samples.csv", "bulk_counts_long.csv")

head(res_dt12$combined_samples, 5)
```

```
##    sample_id condition  batch patient_id timepoint cohort
##       <char>    <char> <char>     <char>    <char> <char>
## 1:       S19   control batch2       P013        T0      A
## 2:       S20   control batch3       P017        T1      A
## 3:       S21   control batch2       P008        T0      A
## 4:       S03   treated batch2       P017        T1      A
## 5:       S04   treated batch3       P044        T0      A
```

```r
head(res_dt12$top_genes_var, 5)
```

```
##         gene variance
##       <char>    <num>
```

```
## 1: GENE_0377 2880.754
## 2: GENE_0425 2422.824
## 3: GENE_0020 1387.326
## 4: GENE_0495 1367.563
## 5: GENE_0272 1350.341
```

```
head(res_dt12$summary_counts, 5)
```

```
##         gene cohort condition mean_count
##       <char> <char>    <char>      <num>
## 1: GENE_0006      A   control   34.00000
## 2: GENE_0020      A   control   30.66667
## 3: GENE_0027      A   control   43.33333
## 4: GENE_0034      A   control   50.66667
## 5: GENE_0044      A   control   70.00000
```

DATA.FRAME VERSION

```
task12_df <- function(cohortA_file = "cohortA_samples.csv",
                      cohortB_file = "cohortB_samples.csv",
                      counts_file  = "bulk_counts_long.csv") {

# Read and align cohorts
A <- read.csv(cohortA_file, stringsAsFactors = FALSE)
B <- read.csv(cohortB_file, stringsAsFactors = FALSE)

all_cols <- union(names(A), names(B))
for (col in setdiff(all_cols, names(A))) A[[col]] <- NA
for (col in setdiff(all_cols, names(B))) B[[col]] <- NA

combined_df <- rbind(A[all_cols], B[all_cols])

# Order by cohort, condition, sample_id
  order_cols <- intersect(c("cohort", "condition", "sample_id"), names(combined_df))
  if (length(order_cols) > 0)
    combined_df <- combined_df[do.call(order, combined_df[order_cols]), ]

# Read counts and join by sample_id
  counts <- read.csv(counts_file, stringsAsFactors = FALSE)
  merged <- merge(counts, combined_df, by = "sample_id")

# Compute variance per gene and select top 100
var_df <- aggregate(count ~ gene, data = merged, FUN = var, na.rm = TRUE)
names(var_df)[2] <- "variance"
var_df <- var_df[order(-var_df$variance, var_df$gene), ]
top_genes <- var_df$gene[1:100]

# Compute per-cohort/per-condition mean counts
filtered <- merged[merged$gene %in% top_genes, ]
summary_counts_df <- aggregate(count ~ gene + cohort + condition,
                               data = filtered,
```

```
                                  FUN = function(x) mean(x, na.rm = TRUE))
names(summary_counts_df)[4] <- "mean_count"

summary_counts_df <- summary_counts_df[order(summary_counts_df$cohort,
                                             summary_counts_df$condition,
                                             summary_counts_df$gene), ]

# Column alignment summary
  alignment_summary <- data.frame(
    Status = c("In both", "Only in A", "Only in B"),
    Count  = c(length(intersect(names(A), names(B))),
               length(setdiff(names(A), names(B))),
               length(setdiff(names(B), names(A))))
  )

return(list(
    combined_samples  = combined_df,
    merged_counts     = merged,
    top_genes_var     = var_df,
    summary_counts_df = summary_counts_df,
    column_check      = alignment_summary
  ))
}

# SHOW RESULTS
res_df12 <- task12_df("cohortA_samples.csv",
                      "cohortB_samples.csv", "bulk_counts_long.csv")
head(res_df12$combined_samples, 5)
```

```
##    sample_id condition  batch patient_id timepoint cohort
## 2        S19   control batch2       P013        T0      A
## 10       S20   control batch3       P017        T1      A
## 5        S21   control batch2       P008        T0      A
## 9        S03   treated batch2       P017        T1      A
## 3        S04   treated batch3       P044        T0      A
```

```
head(res_df12$top_genes_var, 5)
```

```
##          gene variance
## 378 GENE_0377 2880.754
## 426 GENE_0425 2422.824
## 21  GENE_0020 1387.326
## 496 GENE_0495 1367.563
## 273 GENE_0272 1350.341
```

```
head(res_df12$summary_counts_df, 5)
```

```
##        gene cohort condition mean_count
## 1 GENE_0006      A   control   34.00000
## 2 GENE_0020      A   control   30.66667
## 3 GENE_0027      A   control   43.33333
## 4 GENE_0034      A   control   50.66667
## 5 GENE_0044      A   control   70.00000
```

FINAL REVISION

This task links cell type identities to integration clusters and distinguishes whether cells originate from normal or tumor tissue. The two input tables (integration clusters and cell type annotations) are joined by cell to create a combined reference table. From this, cell type counts are summarized per cluster, and the distribution of cell types is compared between normal and tumor samples. A plot is generated to visualize these distributions, and normalized percentages are computed to compare cluster composition independently of total cell numbers.

DATA.TABLE VERSION

```
task13_dt <- function(integration_file = "annotated_GSM3516673_normal_annotated_GSM3516672_tumor_Seurat
                      annotation_file  = "nt_combined_clustering.output.csv") {

# Load libraries
library(data.table)
library(ggplot2)

# Prepare output directory
if (!dir.exists("output")) dir.create("output")

# Read input files
integ_dt <- fread(integration_file)
annot_dt <- fread(annotation_file)

# Clean IDs
integ_dt[, cell := gsub("_X_", "", cell)]

# Merge
merged_dt13 <- merge(integ_dt, annot_dt, by = "cell", sort = FALSE)

# Count per cluster and cell type
count_dt13 <- merged_dt13[, .(count = .N), by = .(integration_cluster, cell_type)]
setorder(count_dt13, integration_cluster, cell_type)

# Summary table: add tissue type and normalized percentages
summary_dt13 <- merged_dt13[, .(count = .N),
                            by = .(integration_cluster, cell_type, sample_type)]
summary_dt13[, total := sum(count), by = .(integration_cluster, sample_type)]
summary_dt13[, percent := round((count / total) * 100, 2)]

setorder(summary_dt13, integration_cluster, sample_type, cell_type)

# Order
summary_dt13[, integration_cluster := factor(integration_cluster,
                                             levels = sort(unique(integration_cluster)))]
summary_dt13[, sample_type := factor(sample_type, levels = c("N", "T"))]

# Plot 1: Distribution plot
plot_counts_dt13 <- ggplot(summary_dt13,
                           aes(x = factor(integration_cluster),
                               y = count,
```

```
                                fill = cell_type)) +
geom_bar(stat = "identity") +
facet_wrap(~sample_type) +
theme_minimal() +
labs(title = "Cell Type Distribution by Cluster and Tissue Type",
         x = "Integration Cluster",
         y = "Number of Cells",
         fill = "Cell Type") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Plot 2: Distribution plot with normalized percentages
plot_percent_dt13 <- ggplot(summary_dt13,
                            aes(x = factor(integration_cluster),
                                y = percent,
                                fill = cell_type)) +
    geom_bar(stat = "identity") +
    facet_wrap(~sample_type) +
    theme_minimal() +
    labs(title = "Cell Distribution by Cluster and Tissue Type (Normalized %)",
         x = "Integration Cluster",
         y = "Percentage (%)",
         fill = "Cell Type") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Save outputs
fwrite(merged_dt13, "output/merged_dt13.csv")
fwrite(count_dt13, "output/count_dt13.csv")
fwrite(summary_dt13, "output/summary_dt13.csv")
ggsave("output/plot_counts_dt13.pdf", plot_counts_dt13, width = 8, height = 5)
ggsave("output/plot_percent_dt13.pdf", plot_percent_dt13, width = 8, height = 5)


  return(list(
    merged_dt13       = merged_dt13,
    count_dt13        = count_dt13,
    summary_dt13      = summary_dt13,
    plot_counts_dt13  = plot_counts_dt13,
    plot_percent_dt13 = plot_percent_dt13
  ))
}

# SHOW RESULTS
res_dt13 <- task13_dt(
  "annotated_GSM3516673_normal_annotated_GSM3516672_tumor_SeuratIntegration.csv",
                      "nt_combined_clustering.output.csv")
head(res_dt13$merged_dt13, 5)
```

```
##                  cell integration_cluster                    cell_type
##                <char>               <int>                       <char>
## 1: X120703408789411.N                   2  Pro-angiogenesis Macrophage
## 2: X120703408793835.N                   8                Non Blood Cell
## 3: X120703409145716.N                   0 Effector/Memory CD4+ T cells
## 4: X120703409339181.N                   1                    MAIT cells
```

```
## 5: X120703409379676.N                  3  Pro-angiogenesis Macrophage
##    sample_type
##         <char>
## 1:           N
## 2:           N
## 3:           N
## 4:           N
## 5:           N
```

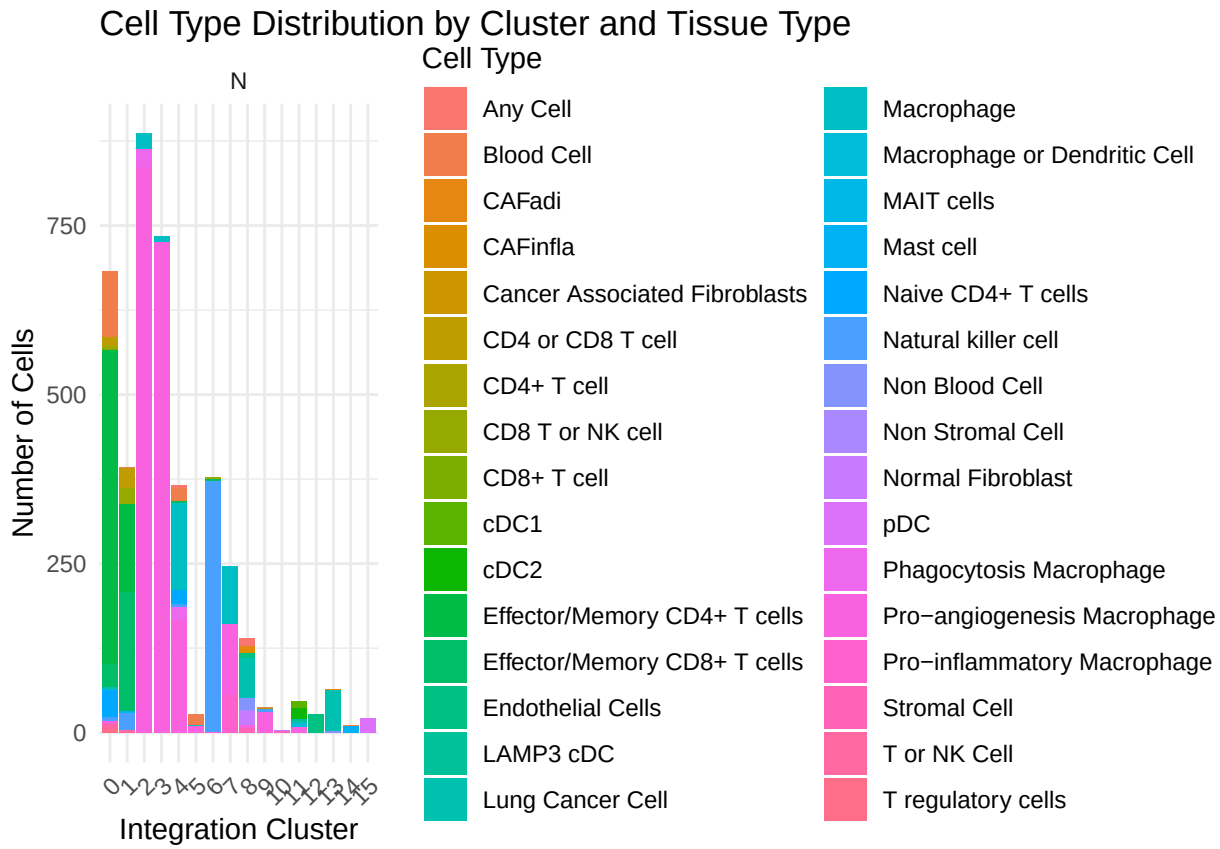
```
head(res_dt13$count_dt13, 5)
```

```
##    integration_cluster         cell_type count
##                  <int>            <char> <int>
## 1:                   0        Blood Cell    97
## 2:                   0 CD4 or CD8 T cell    12
## 3:                   0       CD4+ T cell     3
## 4:                   0  CD8 T or NK cell     3
## 5:                   0       CD8+ T cell     2
```

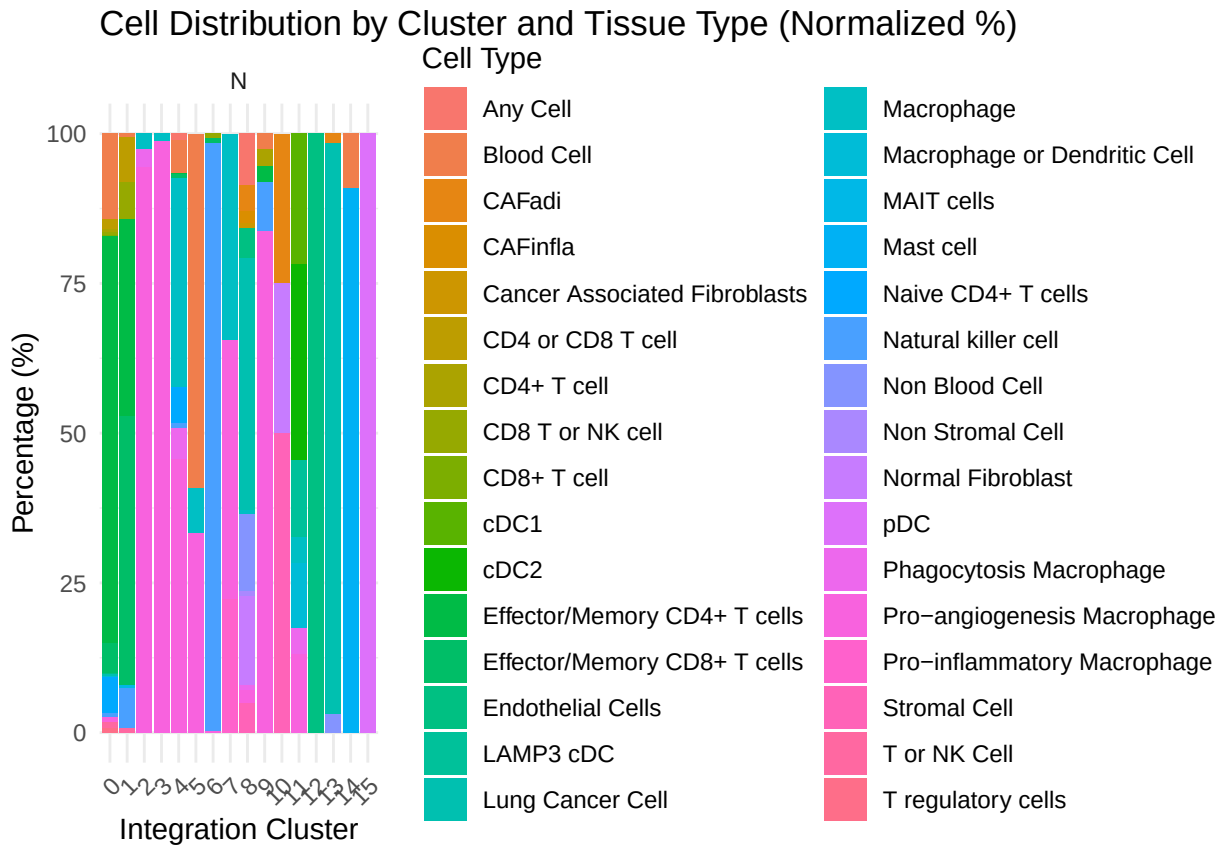
```
head(res_dt13$summary_dt13, 5)
```

```
##    integration_cluster         cell_type sample_type count total percent
##                 <fctr>            <char>      <fctr> <int> <int>   <num>
## 1:                   0        Blood Cell           N    97   682   14.22
## 2:                   0 CD4 or CD8 T cell           N    12   682    1.76
## 3:                   0       CD4+ T cell           N     3   682    0.44
## 4:                   0  CD8 T or NK cell           N     3   682    0.44
## 5:                   0       CD8+ T cell           N     2   682    0.29
```

```
print(res_dt13$plot_counts_dt13)
```




```
print(res_dt13$plot_percent_dt13)
```



DATA.FRAME VERSION

```
task13_df <- function(integration_file = "annotated_GSM3516673_normal_annotated_GSM3516672_tumor_Seurat
                      annotation_file  = "nt_combined_clustering.output.csv") {

# Prepare output directory
if (!dir.exists("output")) dir.create("output")

# Read input files
integ_df <- read.csv(integration_file, stringsAsFactors = FALSE)
annot_df <- read.csv(annotation_file, stringsAsFactors = FALSE)

# Clean IDs
integ_df$cell <- gsub("_X_", "", integ_df$cell)

# Merge
merged_df13 <- merge(integ_df, annot_df, by = "cell")

# Count per cluster
count_df13 <- as.data.frame(table(
    integration_cluster = merged_df13$integration_cluster,
    cell_type = merged_df13$cell_type
  ))
names(count_df13)[3] <- "count"
```

```r
count_df13$count <- as.integer(count_df13$count)
count_df13 <- count_df13[count_df13$count > 0, ]
count_df13 <- count_df13[order(count_df13$integration_cluster,
                               count_df13$cell_type), ]

# Summary with sample_type and normalized %
summary_df13 <- as.data.frame(table(
    integration_cluster = merged_df13$integration_cluster,
    cell_type = merged_df13$cell_type,
    sample_type = merged_df13$sample_type
  ))

names(summary_df13)[4] <- "count"
summary_df13$count <- as.integer(summary_df13$count)
summary_df13 <- summary_df13[summary_df13$count > 0, ]
summary_df13$total <- ave(summary_df13$count,
                          interaction(summary_df13$integration_cluster,
                                      summary_df13$sample_type),
                          FUN = sum)
summary_df13$percent <- round((summary_df13$count / summary_df13$total) * 100, 2)
summary_df13 <- summary_df13[order(summary_df13$integration_cluster,
                                   summary_df13$sample_type,
                                   summary_df13$cell_type), ]
# Order
summary_df13$integration_cluster <- factor(summary_df13$integration_cluster,
                                           levels = sort(unique(
                                             summary_df13$integration_cluster)))
summary_df13$sample_type <- factor(summary_df13$sample_type,
                                   levels = c("N", "T"))

# Plot 1: Distribution plot
  plot_counts_df13 <- ggplot(summary_df13,
                             aes(x = factor(integration_cluster),
                                 y = count,
                                 fill = cell_type)) +
    geom_bar(stat = "identity") +
    facet_wrap(~sample_type) +
    theme_minimal() +
    labs(title = "Cell Type Distribution by Cluster and Tissue Type",
         x = "Integration Cluster",
         y = "Number of Cells",
         fill = "Cell Type") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Plot 2: Distribution plot with normalized percentages
  plot_percent_df13 <- ggplot(summary_df13,
                              aes(x = factor(integration_cluster),
                                  y = percent,
                                  fill = cell_type)) +
    geom_bar(stat = "identity") +
    facet_wrap(~sample_type) +
    theme_minimal() +
    labs(title = "Cell Distribution by Cluster and Tissue Type (Normalized %)",
```

```
         x = "Integration Cluster",
         y = "Percentage (%)",
         fill = "Cell Type") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Save outputs
write.csv(merged_df13, "output/merged_df13.csv", row.names = FALSE)
write.csv(count_df13, "output/count_df13.csv", row.names = FALSE)
write.csv(summary_df13, "output/summary_df13.csv", row.names = FALSE)
ggsave("output/plot_counts_df13.pdf", plot_counts_df13, width = 8, height = 5)
ggsave("output/plot_percent_df13.pdf", plot_percent_df13, width = 8, height = 5)


return(list(
    merged_df13       = merged_df13,
    count_df13        = count_df13,
    summary_df13      = summary_df13,
    plot_counts_df13  = plot_counts_df13,
    plot_percent_df13 = plot_percent_df13
  ))
}

# SHOW RESULTS
res_df13 <- task13_df(
  "annotated_GSM3516673_normal_annotated_GSM3516672_tumor_SeuratIntegration.csv",
                      "nt_combined_clustering.output.csv")
head(res_df13$merged_df13, 5)
```

```
##                 cell integration_cluster                   cell_type
## 1 X120703408789411.N                   2  Pro-angiogenesis Macrophage
## 2 X120703408793835.N                   8               Non Blood Cell
## 3 X120703409145716.N                   0 Effector/Memory CD4+ T cells
## 4 X120703409339181.N                   1                   MAIT cells
## 5 X120703409379676.N                   3  Pro-angiogenesis Macrophage
##   sample_type
## 1           N
## 2           N
## 3           N
## 4           N
## 5           N
```

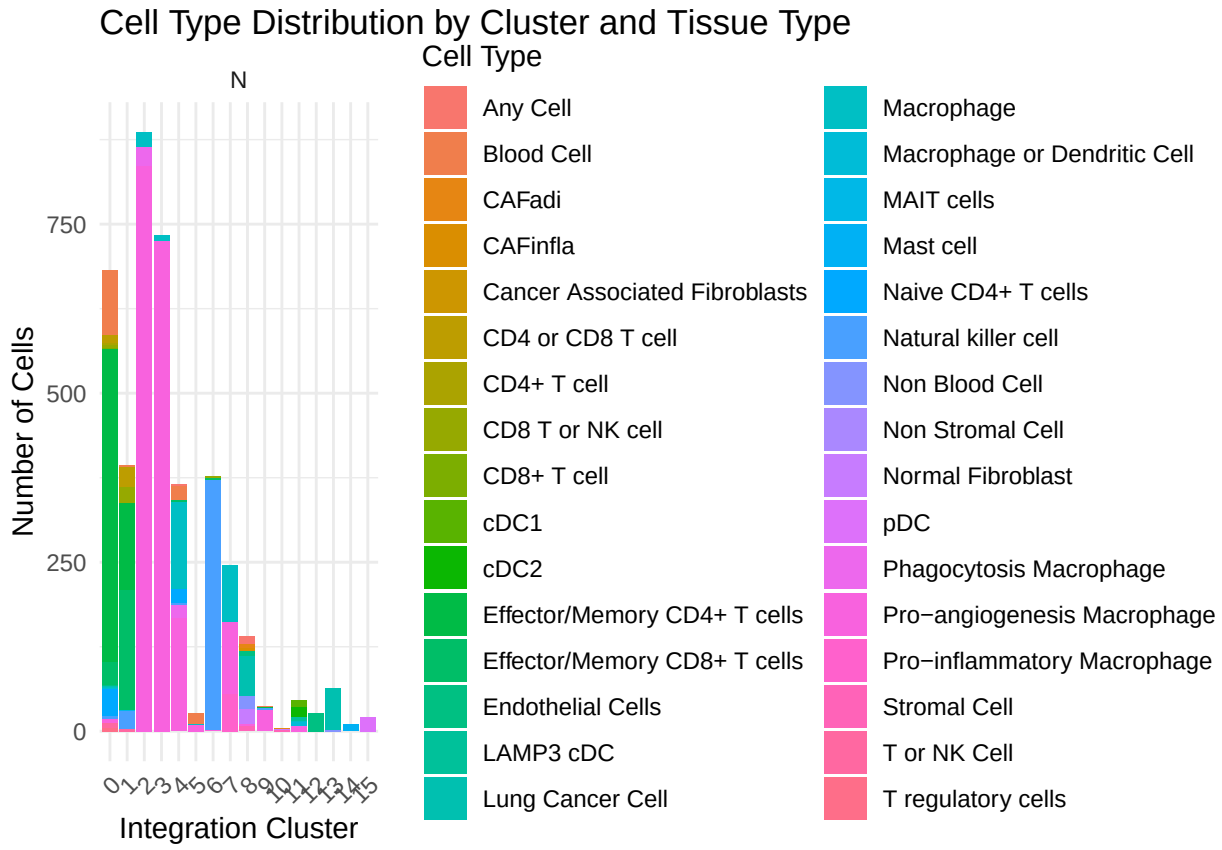
```
head(res_df13$count_df13, 5)
```

```
##     integration_cluster         cell_type count
## 17                    0        Blood Cell    97
## 81                    0 CD4 or CD8 T cell    12
## 97                    0       CD4+ T cell     3
## 113                   0  CD8 T or NK cell     3
## 129                   0       CD8+ T cell     2
```

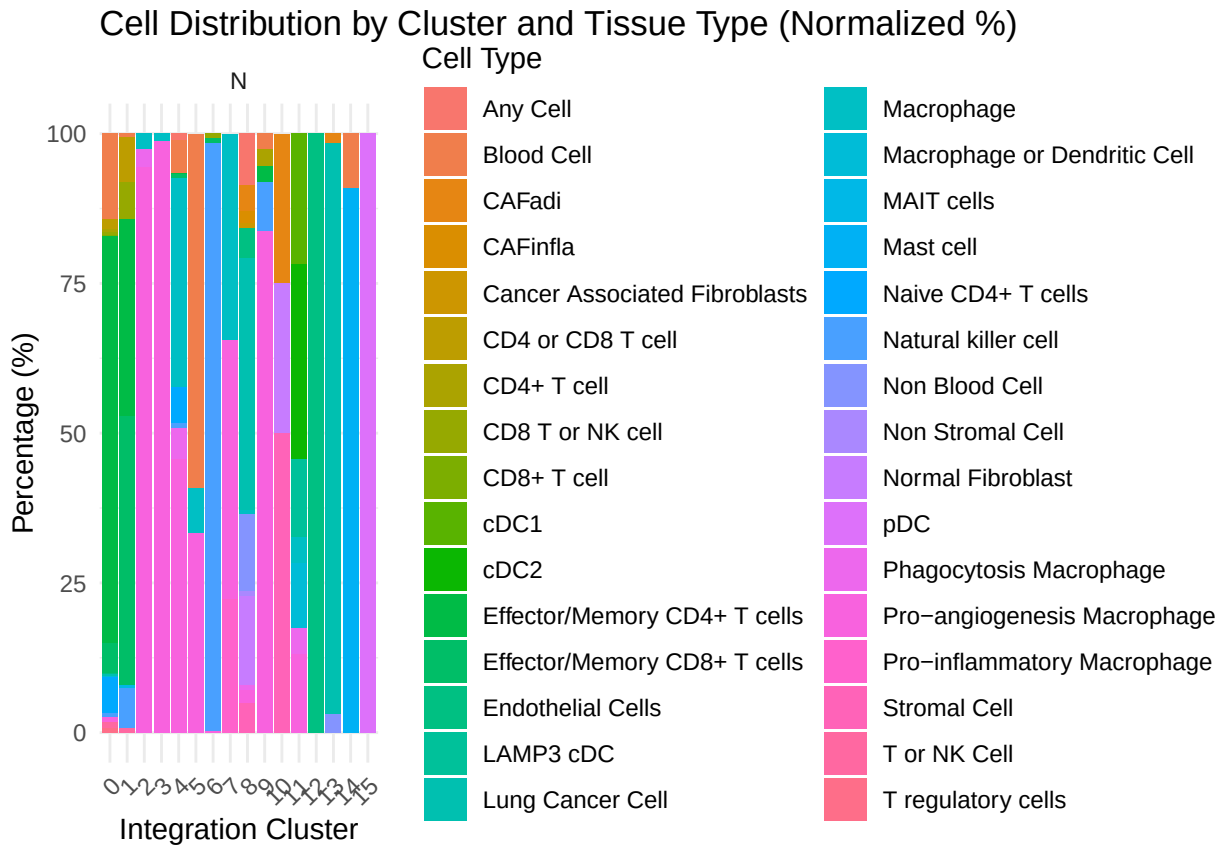
```
head(res_df13$summary_df13, 5)
```

```
##     integration_cluster         cell_type sample_type count total percent
## 17                    0        Blood Cell           N    97   682   14.22
## 81                    0 CD4 or CD8 T cell           N    12   682    1.76
## 97                    0       CD4+ T cell           N     3   682    0.44
## 113                   0  CD8 T or NK cell           N     3   682    0.44
## 129                   0       CD8+ T cell           N     2   682    0.29
```

```
print(res_df13$plot_counts_df13)
```




```
print(res_df13$plot_percent_df13)
```


Cell Distribution by Cluster and Tissue Type (Normalized %)
N
Percentage (%)
100
75
50
25
0
Integration Cluster
Cell Type
Any Cell
Blood Cell
CAFadi
CAFinfla
Cancer Associated Fibroblasts
CD4 or CD8 T cell
CD4+ T cell
CD8 T or NK cell
CD8+ T cell
cDC1
cDC2
Effector/Memory CD4+ T cells
Effector/Memory CD8+ T cells
Endothelial Cells
LAMP3 cDC
Lung Cancer Cell
Macrophage
Macrophage or Dendritic Cell
MAIT cells
Mast cell
Naive CD4+ T cells
Natural killer cell
Non Blood Cell
Non Stromal Cell
Normal Fibroblast
pDC
Phagocytosis Macrophage
Pro−angiogenesis Macrophage
Pro−inflammatory Macrophage
Stromal Cell
T or NK Cell
T regulatory cells